# iOS 7 Tech Talks 2013

San Francisco New York Tokyo Shanghai Berlin London

These are confidential sessions—please refrain from streaming, blogging, or taking pictures

# Integrating iOS 7 Game Technologies

## Game Center and Game Controllers

**Allan Schaffer**
Game Technologies Evangelist
aschaffer@apple.com

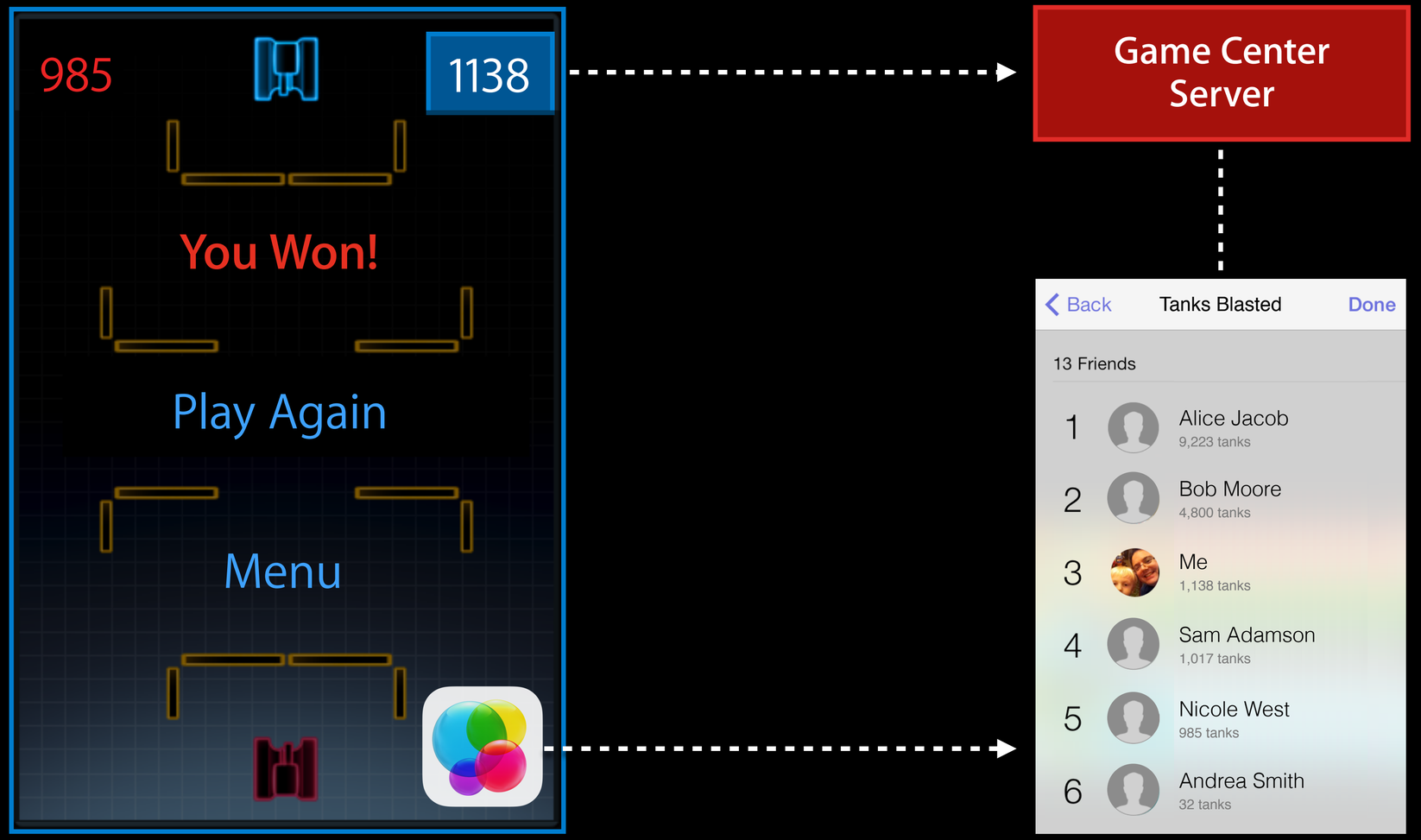
Game Center

Game Controllers

- Friends
- Leaderboards
- Achievements
- Challenges
- Multiplayer

# Three Strategies


# Tie scores and achievements back into gameplay

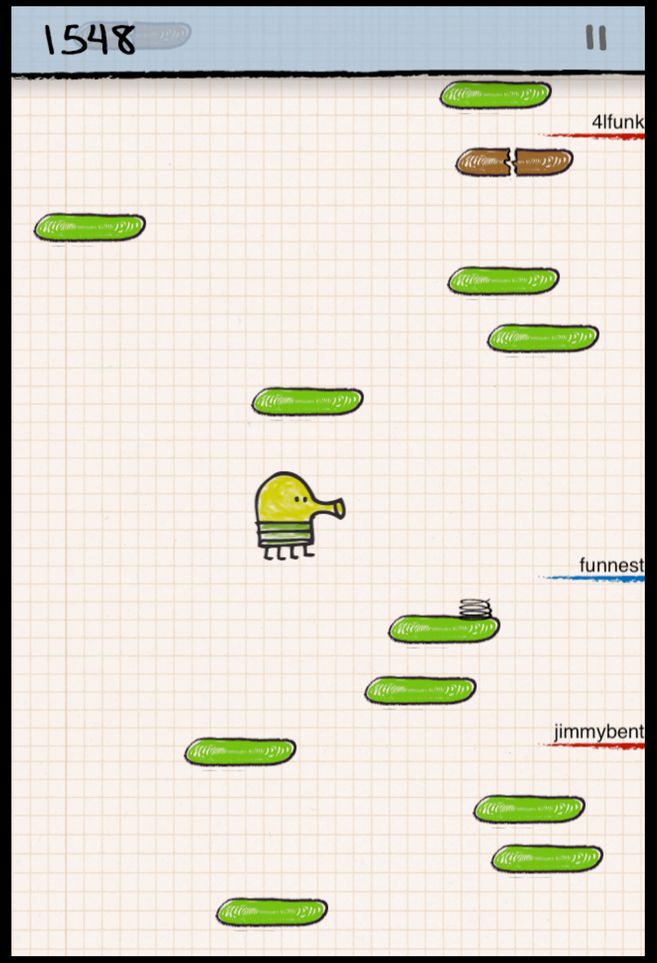

985
1138
You Won!
Play Again
Menu
Game Center
Server
Back
Tanks Blasted
Done
13 Friends
1
Alice Jacob
9,223 tanks
2
Bob Moore
4,800 tanks
3
Me
1,138 tanks
4
Sam Adamson
1,017 tanks
5
Nicole West
985 tanks
6
Andrea Smith
32 tanks

# Tie Scores Back Into Gameplay

## Techniques

- Display friends scores during gameplay
  - Use friends scores as markers
- Display rank among friends


Temple Run




Doodle Jump

# Tie Scores Back Into Gameplay

## How to retrieve scores

- GKLeaderboard
  - Download raw score data from Game Center
    - Score of local player
    - Scores of local player's friends
- GKScore

```
playerID, score (value), rank, date, context, formattedValue, ...
```

- GKPlayer

```
+ loadPlayersForIdentifiers:withCompletionHandler:
– loadPhotoForSize:withCompletionHandler:
NSString *displayName
```

# Retrieving Scores

## Download top 50 friends scores

```
// GKLeaderboard provides access to scores in Game Center
GKLeaderboard *leaderboard = [[GKLeaderboard alloc] init];
```

```
// Configure request: friends' all-time top 50 scores on specified leaderboard
loaderboard.identifier = @"com.company.tankgame.tanksblasted";
leaderboard.timeScope = GKLeaderboardTimeScopeAllTime;
leaderboard.playerScope = GKLeaderboardPlayerScopeFriendsOnly;
leaderboard.range = NSMakeRange(1,50);
```

```
// Fetch scores from Game Center
[leaderboard loadScoresWithCompletionHandler:^(NSArray *scores, NSError *error)
{
    if (scores) // array of GKScores
        [self processScores:scores];
}
```

# Tie Back Into Gameplay

## Going further

- Use the Most Recent Score leaderboard
- Display banners when old score passed
- Use objectives/missions to prime achievements

# Enhancing Player Engagement

## With Game Center

1. Tie scores and achievements back into gameplay

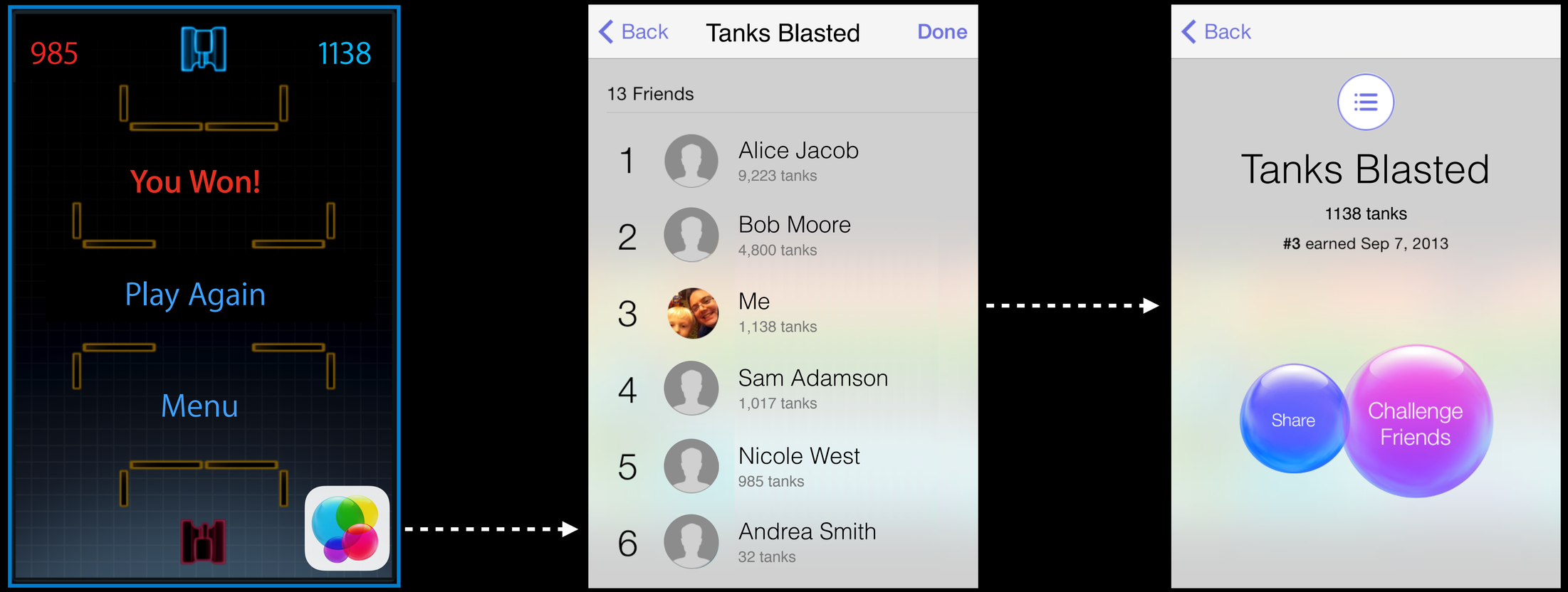

# Bring Challenges Into Game UI

# Challenges

## Quick recap

- “Challenge” a friend to beat your score or earn an achievement
  - Drives competition, engagement, replayability
- Score challenge
  - Based on score earned by challenger
  - Re-challenge issued when challenged player earns better score
  - Continues to re-challenge back and forth
- Achievement challenge
  - Issued from an achievement challenger has earned

# Built-in Challenges UI



# Bring Challenges Into Primary UI

## Techniques

- Offer to send challenge after passing a friend’s score
- Display received challenges in custom UI at launch
- Display received challenge “markers” in-game

# Bring Challenges Back Into Gameplay

## Manage challenges programmatically

- GKLeaderboard
  - Download score data for local player and friends

```
[leaderboard loadScoresWithCompletionHandler:^(NSArray *scores, NSError *error)
```

- GKScore, GKAchievement
  - Issue challenges to friends

```
–challengeComposeControllerWithPlayers:message:completionHandler:
```

- GKChallenge, GKScoreChallenge, GKAchievementChallenge
  - Track recieved challenges

```
+loadReceivedChallengesWithCompletionHandler:
```

# Issuing Challenges

## Players with a lower score

```
-(void) challengeLesserBeings:(int64_t)score inLeaderboard:(NSString*)LBID
{
    // Configure request: friends' all-time top 50 scores   (as shown before)
    GKLeaderboard *leaderboard = [[GKLeaderboard alloc] init];
    ...
    // Fetch scores from Game Center
    [leaderboard loadScoresWithCompletionHandler:^(NSArray *scores, NSError *error) {

        // Filter to get scores less than mine
        NSPredicate *filter = [NSPredicate predicateWithFormat:@"value < %qi", score];
        NSArray *lesserScores = [scores filteredArrayUsingPredicate:filter];

        // Offer to send challenges
        [self presentChallengeWithPreselectedScores:lesserScores];
    }];
}
```

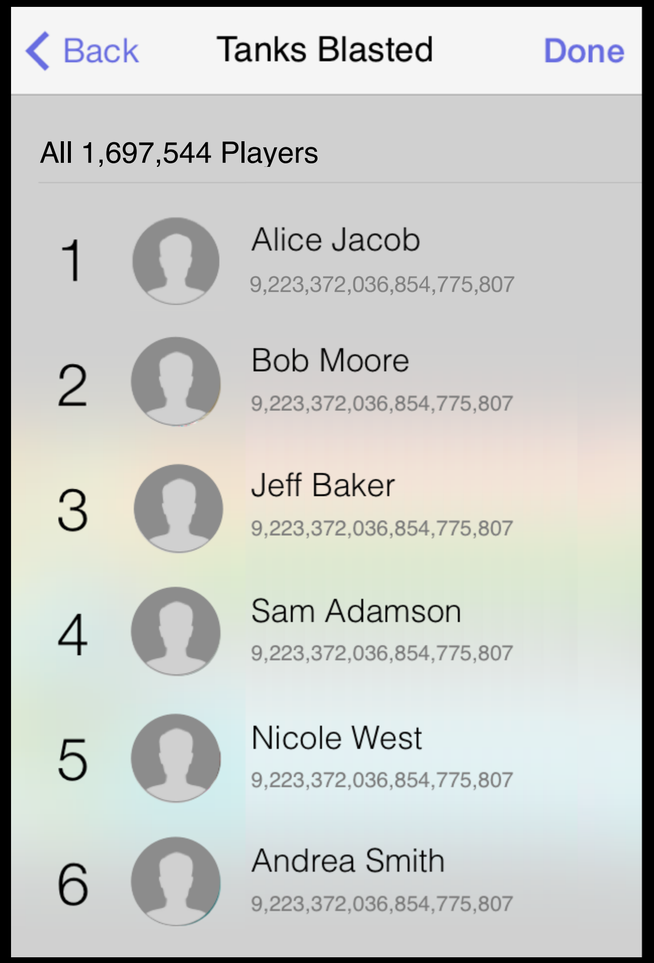
# Enhancing Player Engagement

## With Game Center

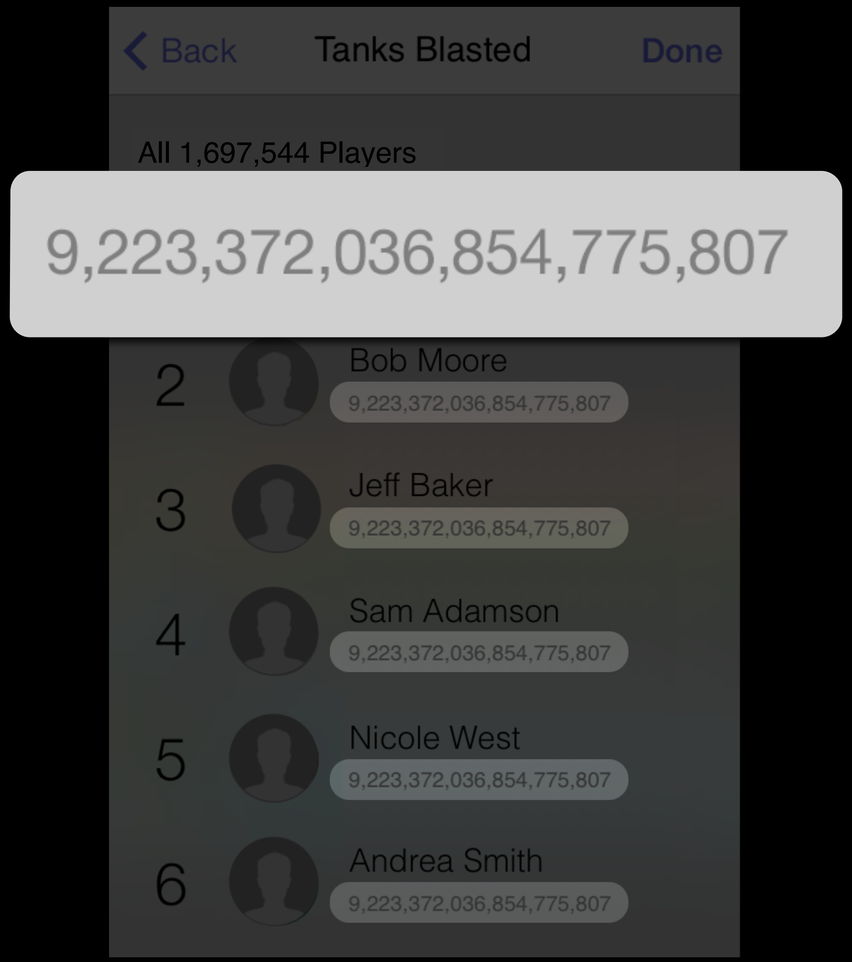
1. Tie scores and achievements back into gameplay
2. Bring Challenges into your UI


# Manage Your Leaderboards

# High Scores


Back
Tanks Blasted
Done
All 1,697,544 Players
1
Alice Jacob
9,223,372,036,854,775,807
2
Bob Moore
9,223,372,036,854,775,807
3
Jeff Baker
9,223,372,036,854,775,807
4
Sam Adamson
9,223,372,036,854,775,807
5
Nicole West
9,223,372,036,854,775,807
6
Andrea Smith
9,223,372,036,854,775,807

# High Scores


Back
Tanks Blasted
Done
All 1,697,544 Players
9,223,372,036,854,775,807
2
Bob Moore
9,223,372,036,854,775,807
3
Jeff Baker
9,223,372,036,854,775,807
4
Sam Adamson
9,223,372,036,854,775,807
5
Nicole West
9,223,372,036,854,775,807
6
Andrea Smith
9,223,372,036,854,775,807

# 9,223,372,036,854,775,807

aka

INT64_MAX

# Limiting Cheating

## Set a score range in iTunes Connect

### Leaderboard Information

| | | |
|---|---|---|
| Leaderboard ID | `com.studioName.myGame.leaderboardID` | |
| Score Submission Type | Best Score ◯ | Most Recent Score ◯ |
| Sort Order | Low to High ◯ | High to Low ◯ |
| Score Range | From ▭ | To ▭ |

# Limiting Cheating

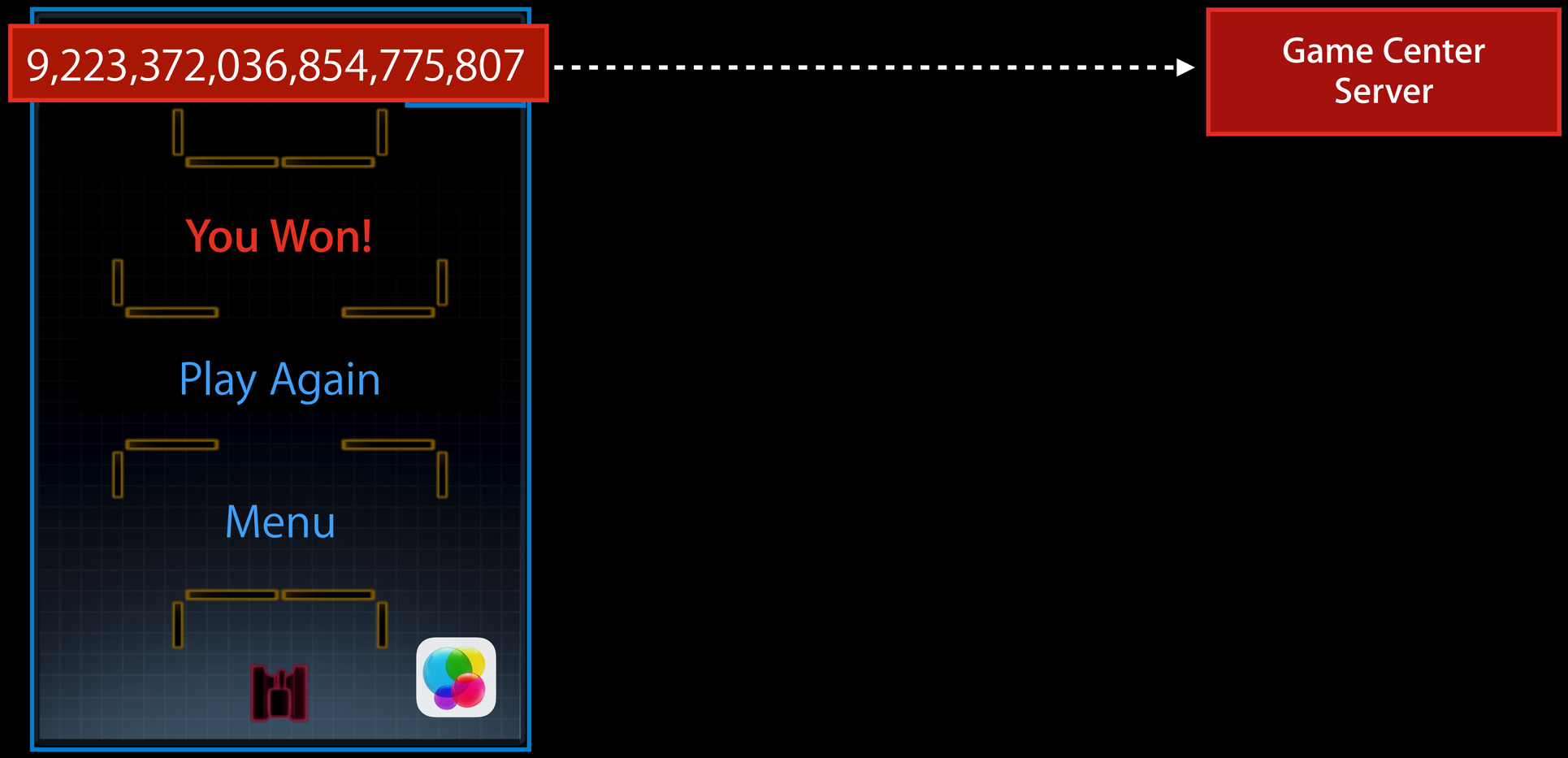
## Set a score range in iTunes Connect

### Leaderboard Information

| | | |
|---|---|---|
| Leaderboard ID | `com.studioName.myGame.leaderboardID` | |
| Score Submission Type | Best Score ○ | Most Recent Score ○ |
| Sort Order | Low to High ○ | High to Low ○ |
| Score Range | From [ ] | To [ ] |

# Limiting Cheating

## Score submissions in iOS 7 are signed



# Limiting Cheating

## Score management in iTunes Connect


- Remove invalid scores and block users
  - View top 100 scores per leaderboard
  - Delete any invalid score
  - Block any user and all their invalid scores
- Now available

# Score Management

## Deleting a score

- Removes score from leaderboard
- User may re-play and ‘earn’ a real score
- No other leaderboards affected

# Score Management

## Blocking a user

- Removes scores from all leaderboards on your app
- Blocks user from posting new scores to your app
- Grouped apps
  - Scores removed from all leaderboards in the group
  - User is blocked from posting new scores for all leaderboards in the group

With great power
comes great responsibility

# Enhancing Player Engagement

## With Game Center

1. Tie scores and achievements back into gameplay
2. Bring challenges up to the surface
3. Manage your leaderboards

Game Center

# Game Controllers

# Game Controllers

# Introduction

## Game controller MFi specification

- For third-party controller developers
  - MFi Program membership required
- Defines hardware requirements
  - Consistent control layouts
  - Fast report rate
  - Pressure sensitive buttons
  - No drift
  - No dead zones

# Introduction

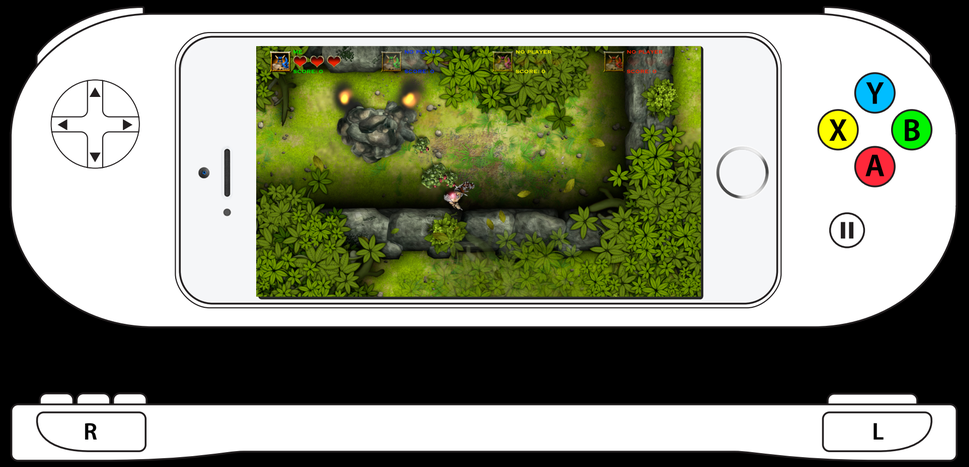
## Game controller framework

- Connect with MFI game controllers
  - Detect controllers
  - Read inputs in-game
- One consistent API for all controllers
  - Focus on making great games

# The Controllers

## Form-fitting standard gamepad

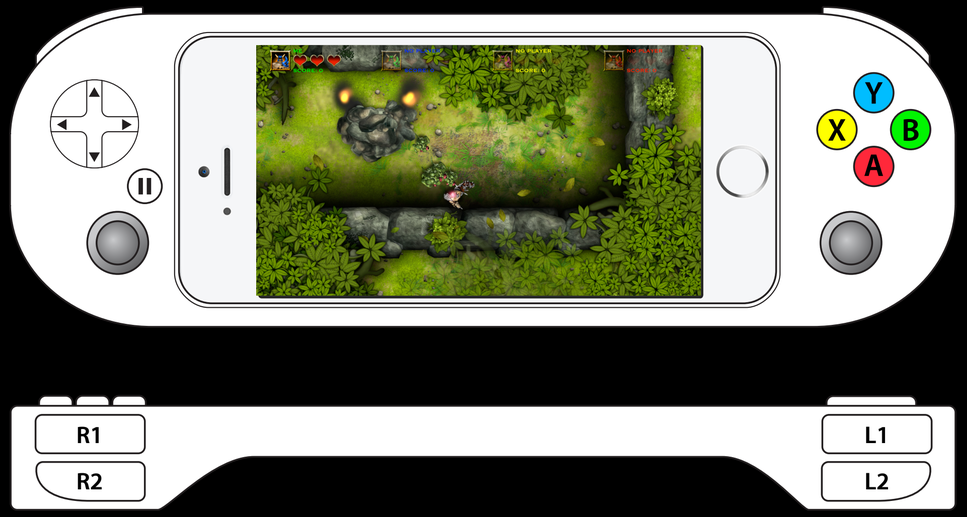
- Form-fitting
  - Physically encases the device
  - User can touch the screen
- Standard gamepad
  - D-pad
  - ABXY
  - Shoulders



# The Controllers

## Form-fitting extended gamepad

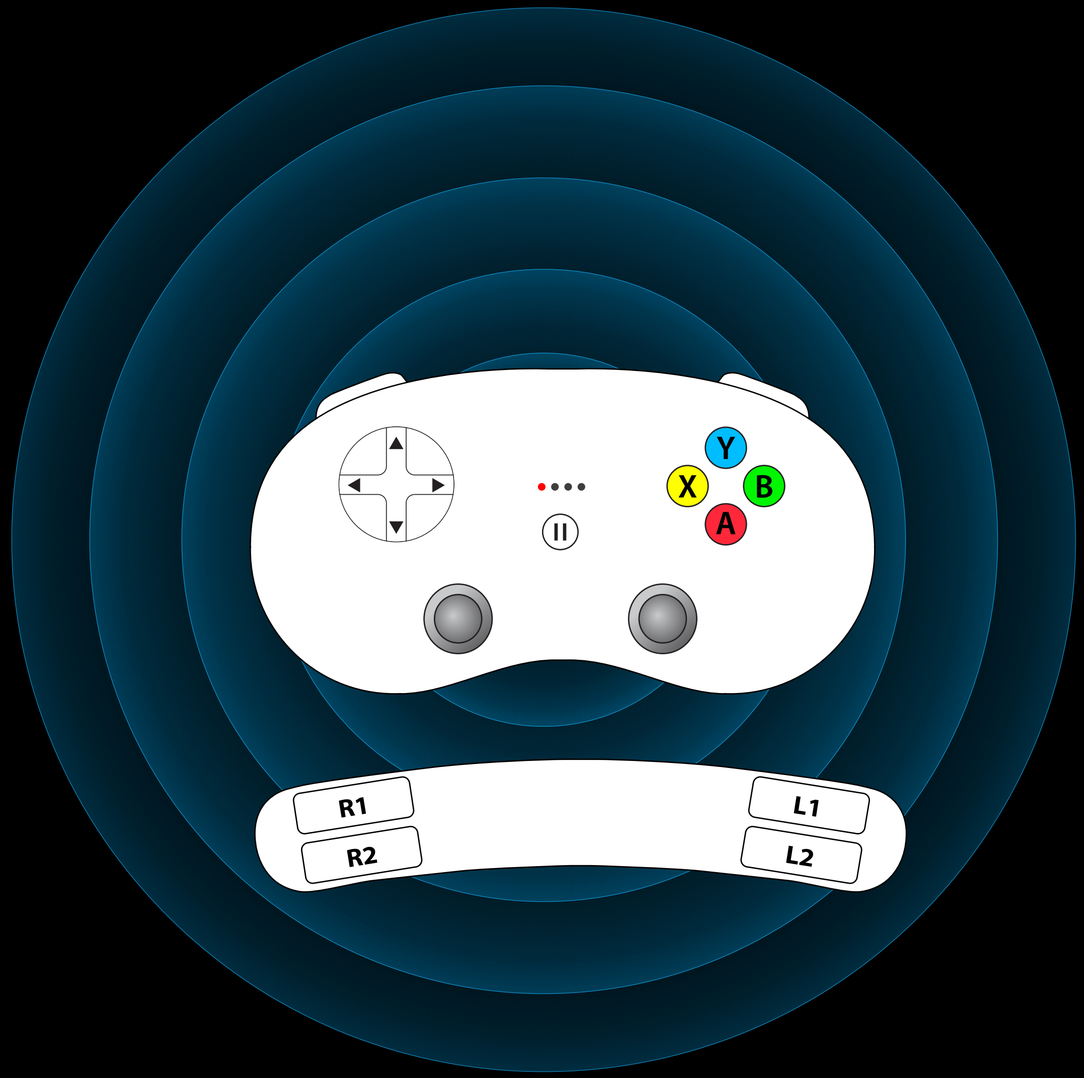
- Form-fitting
  - Physically encases the device
  - User can touch the screen
- Extended gamepad
  - D-pad
  - ABXY
  - Shoulders
  - Thumbsticks
  - Triggers



# The Controllers

## Standalone extended gamepad

- Standalone
  - Not attached to device
  - Wired or wireless
- Extended gamepad
  - D-pad
  - ABXY
  - Shoulders
  - Thumbsticks
  - Triggers



# The Controllers

## Available now


# Using Controllers

# GCController

## Overview

- Main class for working with Controllers
  - Represents a connected game controller
  - Same class for all supported controllers
- Provides
  - Methods for finding controllers
  - Access to physical input data
  - Information about this controller

# Connecting and Disconnecting

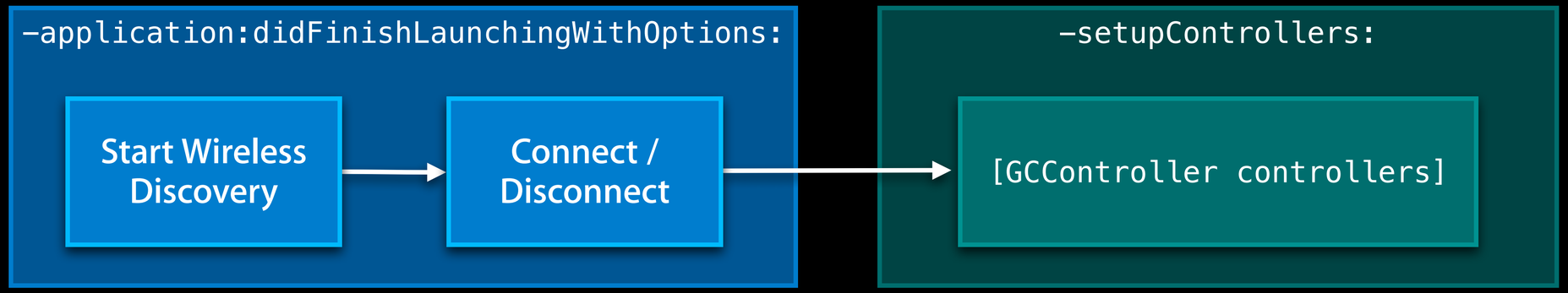
## Main entry point

```
@interface GCController : NSObject
+ (NSArray *)controllers
...
```

- List of currently connected controllers
- Array of GCController instances (empty if none)
- Updated whenever controllers connect or disconnect

# GCController

## Getting connected



# GCController

## Getting connected

```
-(BOOL)application:(UIApplication*)app didFinishLaunchingWithOptions:(NSDictionary*)dict
{
    if ([GCController class])
    {
        NSNotificationCenter* center = [NSNotificationCenter defaultCenter];

        [ center addObserver:self selector:@selector(setupControllers:)
                        name:GCControllerDidConnectNotification object:nil];
        [ center addObserver:self selector:@selector(setupControllers:)
                        name:GCControllerDidDisconnectNotification object:nil];

        [ GCController startWirelessControllerDiscoveryWithCompletionHandler:^{ ... }];
    }
}
```

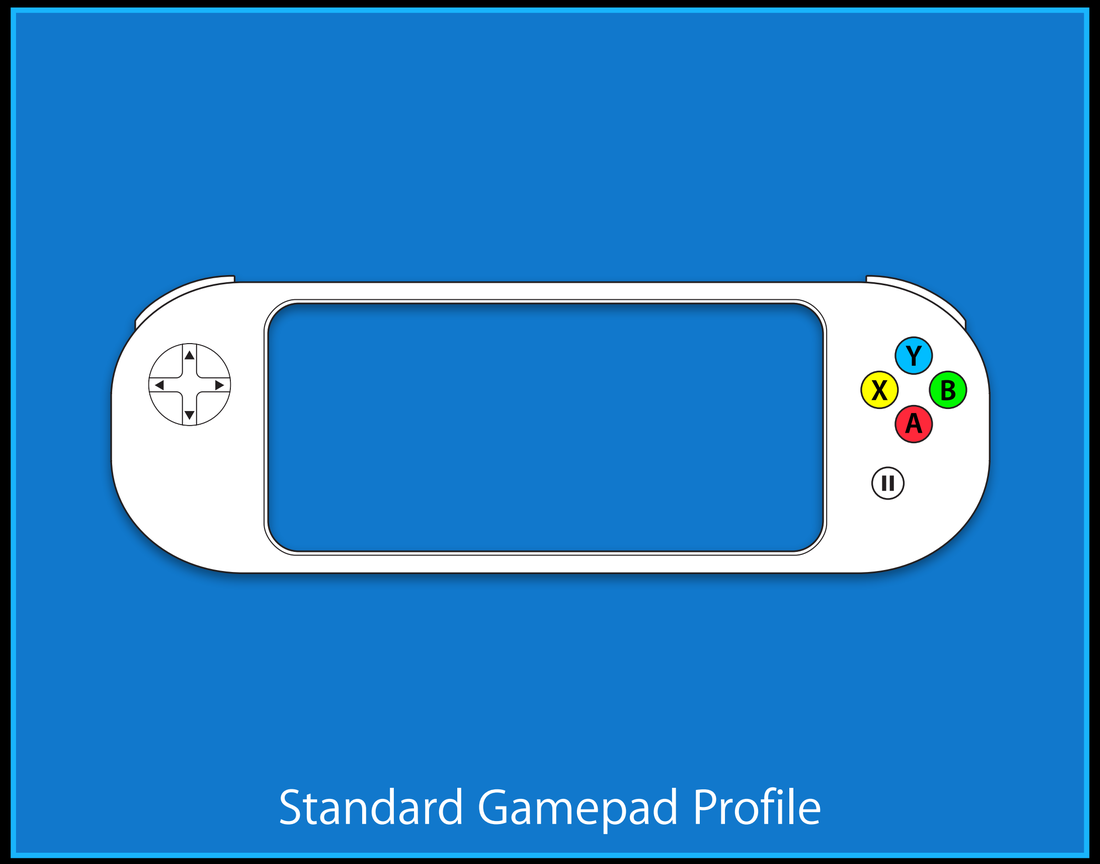
# GCController

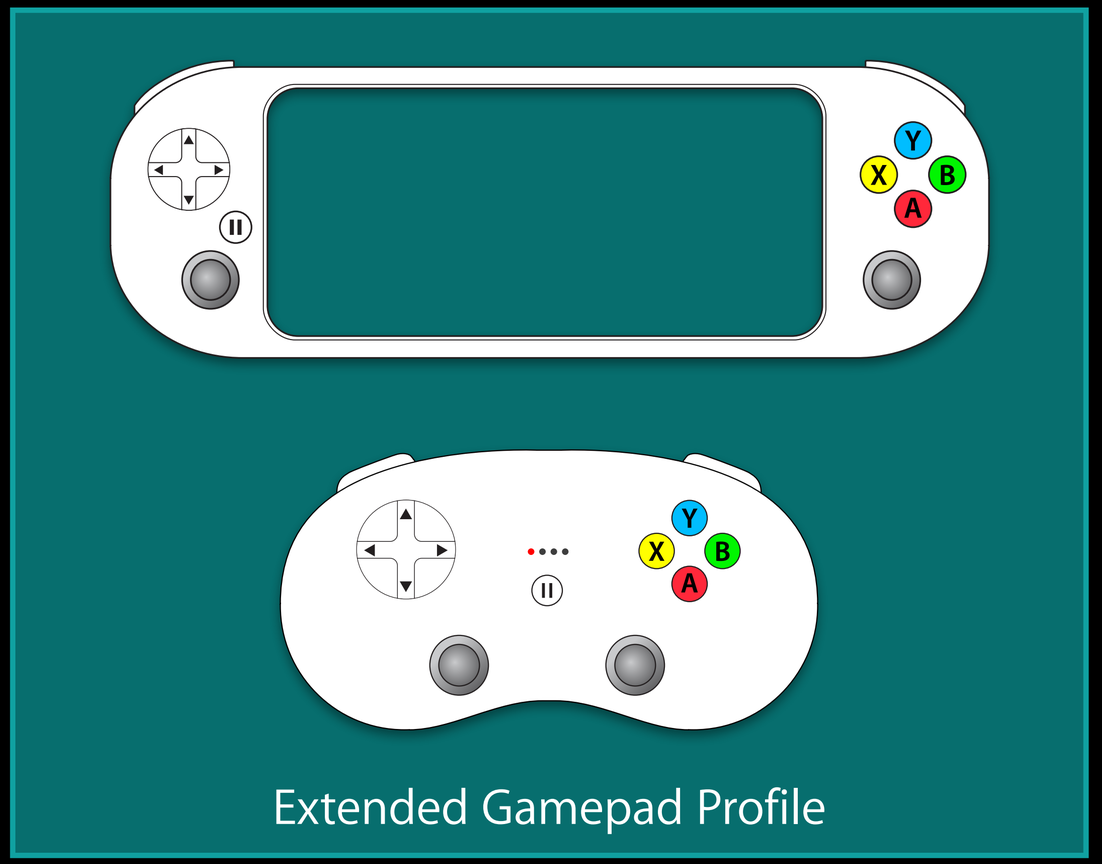
## Getting connected

```
- (void)setupControllers:(NSNotification *)notification
{
    // Get array of connected controllers
    self.controllerArray = [GCController controllers];

    if ([self.controllerArray count] > 0)
        // Found controllers
    ...
}
```

# Reading Controller Input

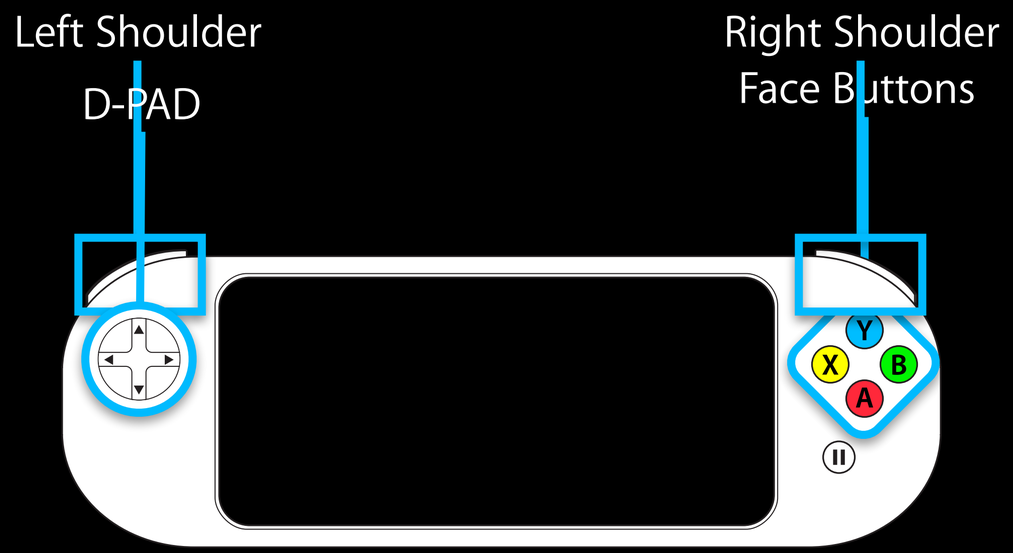
# Controller Profiles



# Controller Profiles

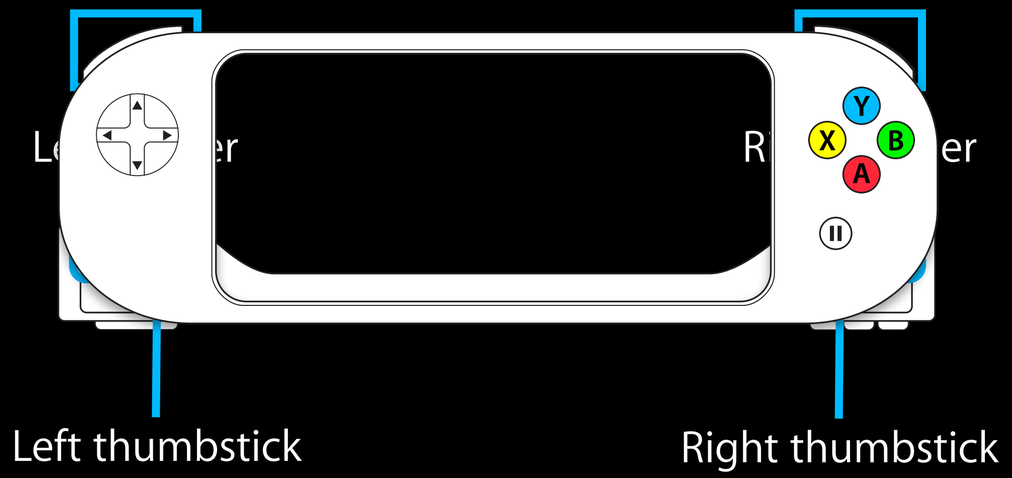
## Standard gamepad profile




| Property Name | Type |
| --- | --- |
| buttonA | GCButtonInput |
| buttonB | |
| buttonX | |
| buttonY | |
| leftShoulder | GCButtonInput |
| rightShoulder | |
| dpad | GCDirectionPad |

# Controller Profiles

## Extended gamepad profile




| Property Name | Type |
|---|---|
| buttonA | GCButtonInput |
| buttonB | |
| buttonX | |
| buttonY | |
| leftShoulder | GCButtonInput |
| rightShoulder | |
| dpad | GCDirectionPad |
| leftThumbstick | GCDirectionPad |
| rightThumbstick | |
| leftTrigger | GCButtonInput |
| rightTrigger | |

# Element Types

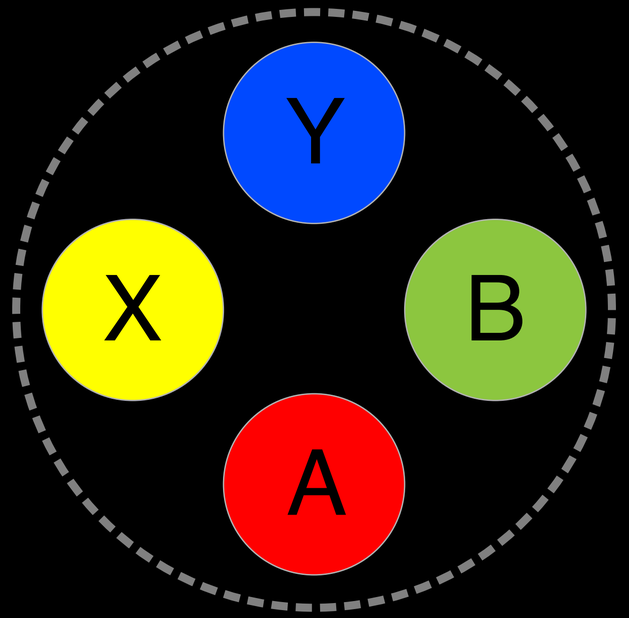
## GCControllerButtonInput

- Classic button state

```
BOOL pressed;     // Whether button is pressed
```

- Buttons are also pressure sensitive

```
float value;      // Amount of pressure (analog)
                  // Normalized from 0.0 to 1.0
```



# Element Types

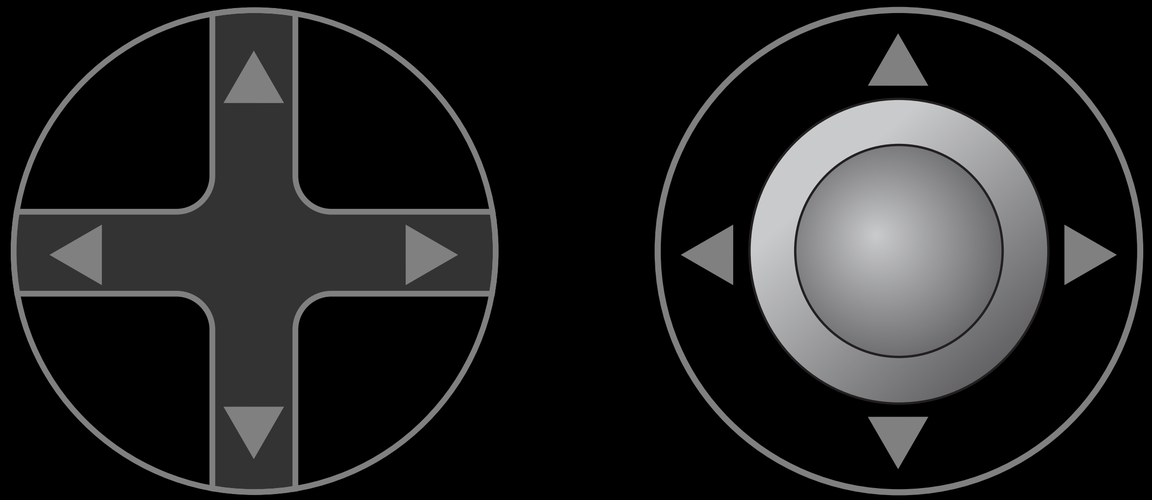
## GCControllerDirectionPad

- Treated as four buttons

```
GCControllerButtonInput *up, *down, *left, *right;
```

- Or as two axes

```
GCControllerAxisInput *xAxis, *yAxis;
```



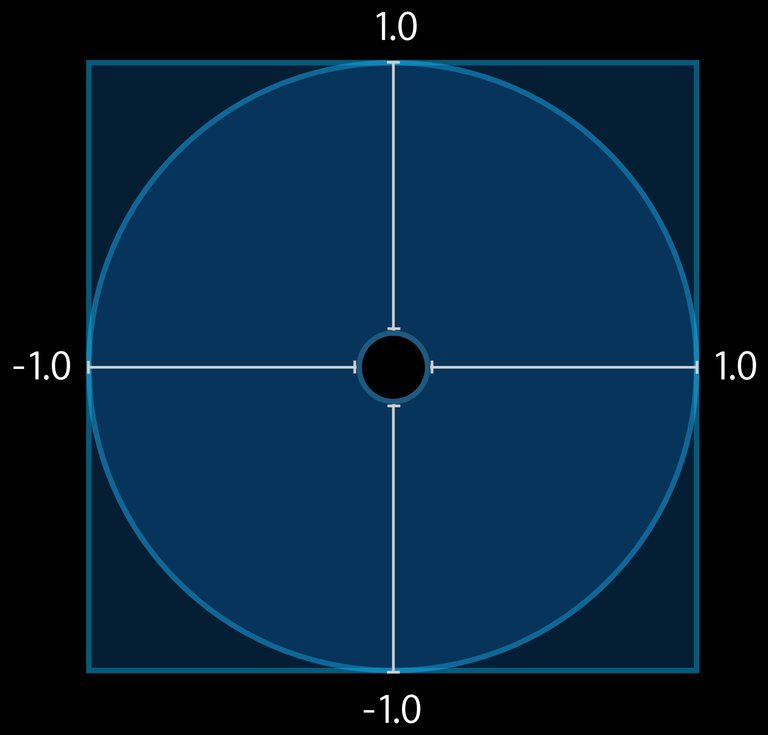
# Element Types

## GCControllerAxisInput

- Measures movement along a particular axis

```
float value;     // Normalized from –1.0 to 1.0; 0.0 is neutral
```

- Unit circle in unit square
- Automatic calibration
- Automatic dead-zone handling



# Reading Element Values

## Three techniques

- Read elements directly
  - Poll controller for inputs
- Register a value change callback
  - Detecting when values change
- Take snapshot of entire controller state
  - For capturing all elements simultaneously

# Reading Element Values

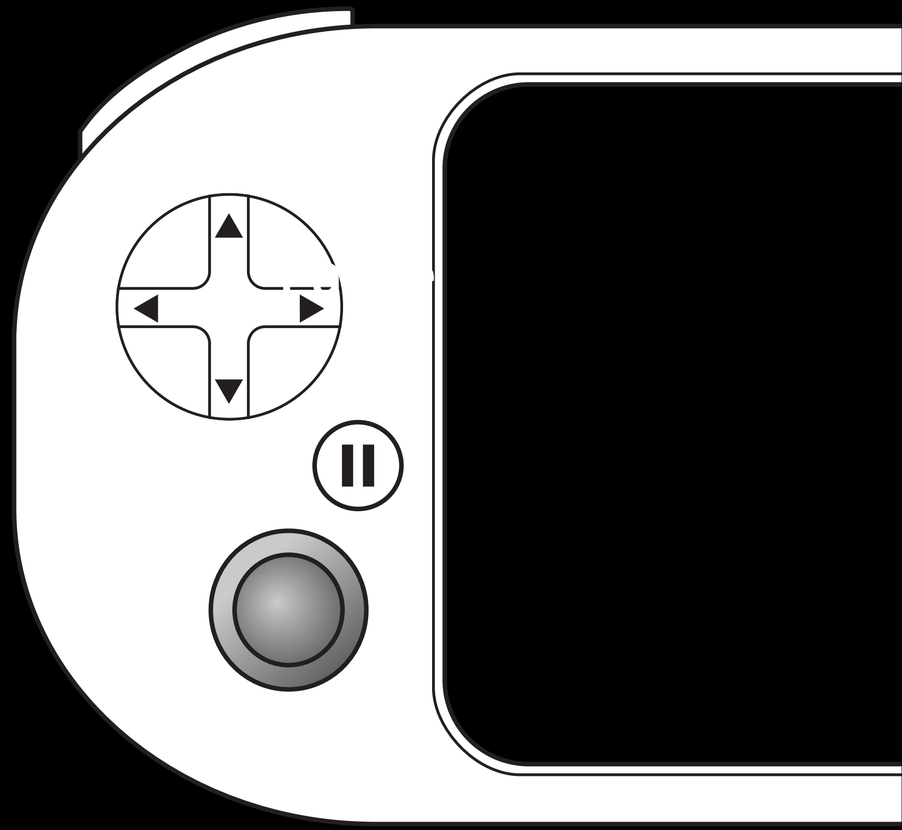
## Polling example

```
-(void)readControls
{
    // Using Extended Gamepad controller
    GCExtendedGamePad *profile = self.myController.extendedGamepad;

    // Take actions for triggers
    if (profile.leftTrigger.isPressed)
        [self launchMissiles];
   if (profile.rightTrigger.isPressed)
        [self fireLasersAtRate:profile.rightTrigger.value];

    // Apply thrust based on Y value of thumbstick
    [self applyThrust: profile.leftThumbstick.yAxis.value];
}
```

# Additional Controls

## Pause button

- Every controller includes Pause button
- Handling required
  - If game supports controllers
- Treat as a toggle
  - Active → Pause
  - Paused → Active
- Consider UI state
  - Inactive → Inactive



# Additional Controls

## Pause button example

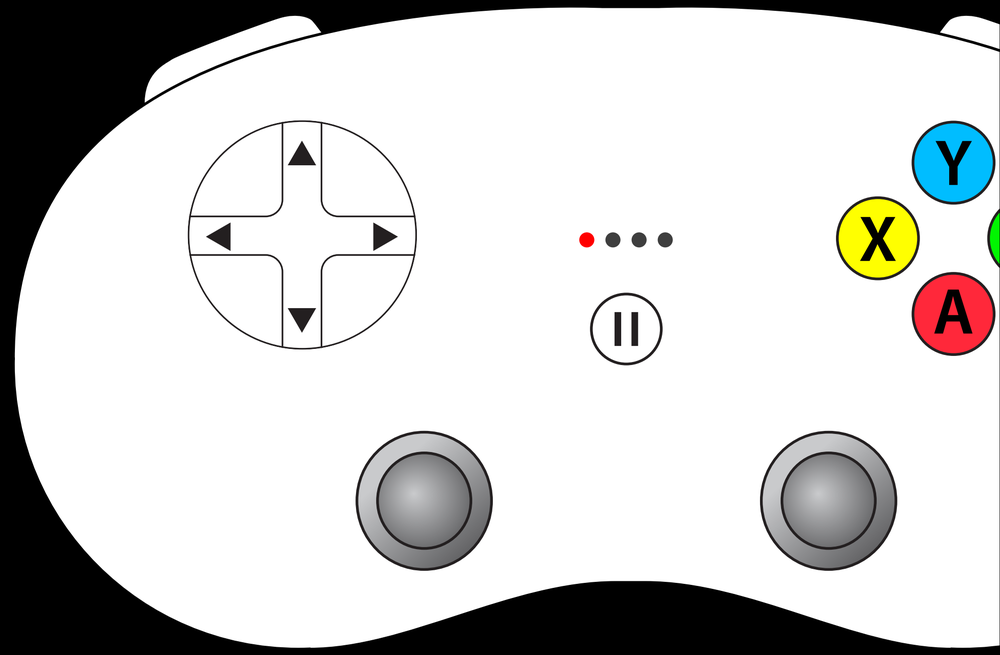
```
- (void)setupControllers
{

    // ...
```

```
    // Add Pause Handler
    self.myController.controllerPausedHandler = ^(GCController *controller)
    {
        // Pause button pressed
        [self togglePauseResumeState];
    }
}
```

# Additional Controls

## Player indicator LEDs

- Controllers may include player indicators
  - Four LEDs
  - API to set/get
  - Persistent
- Always set LEDs
  - For controllers being used
  - Player index



# Designing for Controllers

# Think through your controls

- Touch and motion
  - Ideal for direct interactions
  - i.e., taps, gestures, swipes
- Controller
  - Good for precise or sequenced input
  - i.e., tight moves, actions
- Touch and form-fitting controller together
  - Fine-grain controls plus direct manipulation

# Best Practices

## For games supporting controllers

- Can not require a controller
  - Game controllers are optional
  - Game must work without controller
- Support all profiles and form factors
  - Standard gamepad, extended gamepad
  - Form-fitting, Stand-alone
- Support the Pause button
  - Pause if active, go active if paused

# Best Practices

## Follow standard conventions

- When connected
  - Move to controller-based input
  - Remove on-screen virtual controls
  - Illuminate player indicator
- When disconnecting
  - Pause gameplay
  - Return to regular controls

# Designing for Controllers

## Closing thoughts

Multi-Touch and Motion

+

Game Controllers

# More Information

**Allan Schaffer**
Game Technologies Evangelist
aschaffer@apple.com

**Apple Developer Forums**
http://devforums.apple.com/

**Developer Documentation**
http://developer.apple.com/library/